From Jeremy :

I. EUROPA as flight software

There are a variety of barriers to using EUROPA as flight software. I have a great list of issues:

1. serial reusability If we ever re-start a process, it might not work correctly unless its static variables are explicitly re-set to initial values.

2. sequestering of heap memory If a VxWorks task terminates in such a way that it does not call its constructors, the heap memory it used may remain unavailable to other tasks until VxWorks is re-booted. If each task has its own allocator, the allocator can be re-initialized on startup.

3. running out of heap memory We might be surprised to find that the aggregate demand for heap memory is more than is available to the tasks. I don't know how VxWorks behaves when this happens, but I expect the worst: a system crash. We could avert this by setting an upper bound on the dynamic memory each process can use, and enforce it by supplying our own allocators. In any event, programs should be tested for memory leaks, those memory leaks being removed, and the programes should be designed to minimize use of dynamic memory.

4. preventing infinite recursion Unless tail recursive with a smart enough compiler, infinite or excessively deep recursion will result in a stack overflow. Again, I don't know how VxWorks handles this, but I expect a system crash. The most severe way to handle this is by forbidding recursion, but putting depth bounds on recursive algorithms might be sufficient.

5. Bounds-checking all array references This is easy to do but tedious. Replacing C arrays with STL containers such as std::vector replaces (the risk of) illegal memory references with (the risk of) forbidden iterator operations (advance past end and dereference end).

6. Removing exceptions This is tedious and not at all easy to do.

7. Other requirements placed on L2 No STL No anonymous unions No pointer arithmetic No use of inline keyword No operator overloading

II. EUROPA as Mission Operations tools

There are a couple of things coming down the pike that will be useful to have. EUROPA can't generate resource profiles for activities with linear resource impacts (when plans have fixed start times, end times and durations). This capability will come in handy as we do more work for Mission Operations, both unmanned and manned.

While reasoning capability for preferences is available in EUROPA (e.g. the minimum perturbation heuristic, start time preferences, etc) modeling support for this is lacking. Packaging EUROPA and the Ensemble tools for "single-delivery" is a worthy goal as well. It should be possible to deliver a capability to mission operations that: builds out of the box, has one (and only one) interface and one (and only one) modeling language for the user. Based on my experience with the Ensemble/EUROPA integration today, this is not quite there yet, but is pretty close.

III. EUROPA Tools

I am leaving aside issues surrounding ANML, which you're probably familiar with already.

Languages (translations): We have a profusion of languages: NDDL, ANML, "passive violations" NDDL and Activity Dictionary, at a minimum. We could also toss in XiDDL modeling for IDEA. This is a lot of languages; if we consider the user who has to learn several of these languages for their applications, it's not ideal. Automating the translation of these languages to the maximum extent possible is important, as well as deciding what the user actually builds models in. Modeling tools: The use of the Ensemble tools for development as well as operations is something that would augment the user experience. In addition to that, there is no user support for visualizing models as well as visual modeling. As a means of both augmenting tools such as PRIDE for SAVH, but also for other users (e.g. builders of ADs), this is something to keep in mind. Finally, as you build other modeling support tools that supplant PlanWorks, it is worth keeping PlanWorks design pros and cons in mind. Each "pane" of PlanWorks can (and should!) be a different tool, and tool interaction ala Ensemble is a way to accomplish the navigation features we tried to build into PlanWorks. Finally, ALL of these tools should be available to developers and end users.

IV. More technical EUROPA issues

I am leaving aside the issue of heuristics, which Dave Smith is already on top of.

Dynamic EUROPA and EUROPA: these aren't exactly the same technology, but they are quite closely related; should they be bundled together? I'd also like to know what the differences in the technology are; in part this was motivated by a question about releasing DynamicEuropa as another product.

Constraint violations and passive violation reporting: the passive violation reporting is now limited to "unary resources"; it could be extended to handle other constraints. Is there value in doing this?

Flexible envelopes: For linear resources this is mostly on Paul and my shoulders, but if we get the theory right we could implement. For piecewise constant, some theory work could be put into practice for search control, and some more theory work on search control might be valuable.

EUROPA local search: promotion of tighter integration of EUROPA and Aspen.